The Fork-Join Real-Time Task Model

Martin Stigge, Pontus Ekberg and Wang Yi
Uppsala University, Sweden
Email: {martin.stigge | pontus.ekberg | yi}@it.uu.se

***Abstract*—Hard real-time task models have evolved from periodic models to more sophisticated graph-based ones like the Digraph Real-Time Task Model (DRT) [1]. These models have in common that tasks are *sequential* in nature and do not allow for *forking* structures, modeling job releases that occur in parallel within the same task. To capture these, we present a task model that extends the DRT model with the possibility of forking and joining release paths. We are developing an exact schedulability test for EDF on uniprocessor systems with a pseudo-polynomial bound of its runtime.**

I. Task Model and Problem Statement

A fork-join real-time (FJRT) task system $\tau = \{T_1, \ldots, T_N\}$ consists of N independent tasks. A task T is represented by a directed hypergraph[1] $G(T)$ with both vertex and edge labels. The vertices v_i represent the types of all the jobs that T can release and are labeled with ordered pairs $\langle e(v_i), d(v_i)\rangle$ of non-negative integers, denoting worst-case execution-time demand $e(v_i)$ and relative deadline $d(v_i)$ of the corresponding job. The hyperedges of $G(T)$ represent the order in which jobs generated by T are released. A hyperedge (U, V) is either a *sequence edge* with U and V being singleton sets of vertices, or a *fork edge* with U being a singleton set, or a *join edge* with V being a singleton set. In all cases, the edges are labeled with a non-negative integer $p(U, V)$ denoting the minimum job inter-release separation time. Note that this contains the DRT model as a special case if all hyperedges are sequence edges.

As an extension of the DRT model, an FJRT task system releases independent jobs, allowing to define concepts like *utilization* $U(\tau)$ and *demand bound function* just as before (in e.g. [1]).

Semantics: A task executes by following a path through the hypergraph, triggering releases of associated jobs each time a vertex is visited. Whenever a fork edge $(\{u\}, \{v_1, \ldots, v_m\})$ is taken, m independent paths starting in v_1 to v_m, respectively, will be followed in parallel until joined by a corresponding join edge. In order for a join edge $(\{u_1, \ldots, u_n\}, \{v\})$ to be taken, all jobs associated with vertices $u_1, \ldots, u_n$ must have been released and enough time must have passed to satisfy the join edge label. Forking can be nested, i.e., these m paths can lead to further fork edges before being joined. Note that meaningful models have to satisfy structural restrictions which we skip for space reasons, e.g., each fork needs to be joined by a matching join, and control is not allowed to "jump" between parallel sections. Consider the example in Figure 1.

The main objective of this work is to develop an efficient method for analyzing EDF schedulability for the above model. Thus, we seek to provide a proof to the following conjecture.

Conjecture I.1. *For an FJRT task system τ with $U(\tau) \leqslant c$ for some constant $c < 1$, feasibility can be decided in pseudo-polynomial time.*

Similar results have been shown for simpler task models like the DRT model [1].

[1] A hypergraph generalizes the notion of a graph by extending the concept of an edge (u, v) between two vertices u and v to hyperedges (U, V) between two sets U and V of vertices.

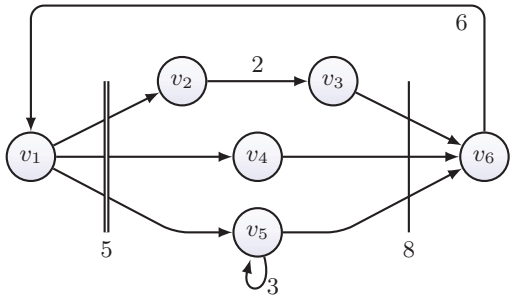

Fig. 1. Example FJRT task. The fork edge is depicted with an intersecting double line, the join edge with an intersecting single line. All edges are annotated with minimum inter-release delays $p(U, V)$. The vertex labels are omitted in this example. Assuming that vertex v_i releases a job of type J_i, a possible job sequence containing jobs with their types and absolute release times is $\sigma = [(J_1, 0), (J_2, 5), (J_5, 5), (J_4, 6), (J_3, 7), (J_5, 8), (J_6, 16), (J_1, 22)]$.

II. Solution Approach

Demand Bound Function: A demand bound function $dbf_T(t)$ for an FJRT task T can be defined as usual as the maximum cumulative execution time requirement of jobs with both release times and deadlines within any interval of length t. Using this, a precise schedulability test for EDF can be based on the following proposition (and EDF's optimality).

Proposition II.1. *An FJRT task system τ is preemptive uniprocessor feasible if and only if*

$$\forall t \geqslant 0 : \sum_{T \in \tau} dbf_T(t) \leqslant t. \tag{1}$$

Demand Tuples: For an FJRT task T without fork and join edges, $dbf_T(t)$ can be evaluated by traversing its graph $G(T)$ using a path abstraction called demand tuples [1]. We can extend this method to the new hyperedges by a recursive approach. Starting with "innermost" fork/join parts of the hypergraph, the tuples are merged at the hyperedges and then used as path abstractions like before. It can be shown that this method is efficient.

Utilization: Just computing $dbf_T(t)$ does not suffice for evaluating Condition (1) since it is also necessary to know which interval sizes t need to be checked. As for the DRT model [1], a bound can be derived from the task set's utilization $U(\tau)$. It turns out that an efficient way of computing $U(\tau)$ is surprisingly difficult to find and is still work in progress. The difficulty comes from parallel sections in the task with loops of different periods which, when joined, exhibit the worst case behavior in very long time intervals of not necessarily polynomial length.

References

[1] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The Digraph Real-Time Task Model," in *Proc. of RTAS 2011*, pp. 71–80.